Polynomial Calculator

Vlad-Cristian Buda 30423

Table of Contents (no numbering)

1. Objective
2. Analysis of the problem
   1. Assumptions
   2. Modeling
   3. Use cases
   4. Errors
3. Design
4. Implementation
   1. Class Monom
   2. Class Polinom
   3. Class Operation
   4. Class GUI
5. Testing
6. Results
7. Conclusions
8. Biobliography

1. Objective

The main objective of this project is to design and develop a Java program specialized in processing polynomial expressions of one variable and integer coefficients.

In order to obtain this functionality, the main problem can be decomposed into several steps needed to reach the goal: obtain a proper representation of a monomial and therefore the polynomial itself; correctly design the methods used to perform operations on polynomials (addition, substraction, multiplication, division, differentiation, integration); design a friendly and intuitive graphical user interface for the user to easily take advantage of the functions provided by this program.

2. Analysis of the problem

The solution of the problem should be able to correctly process polynomials, i.e. perform the following operations: addition, substraction, multiplication, division, differentiation and integration of one or two polynomial expressions. First of all, in order to do that, a correct representation of the polynomials must be proposed and the methods for performing the operations should respect the rules imposed by Mathematics. Another aspect worth taking into consideration is the way the user inputs the polynomials which leads us to the next topic regarding the overall analysis of the problem.

a) Assumptions

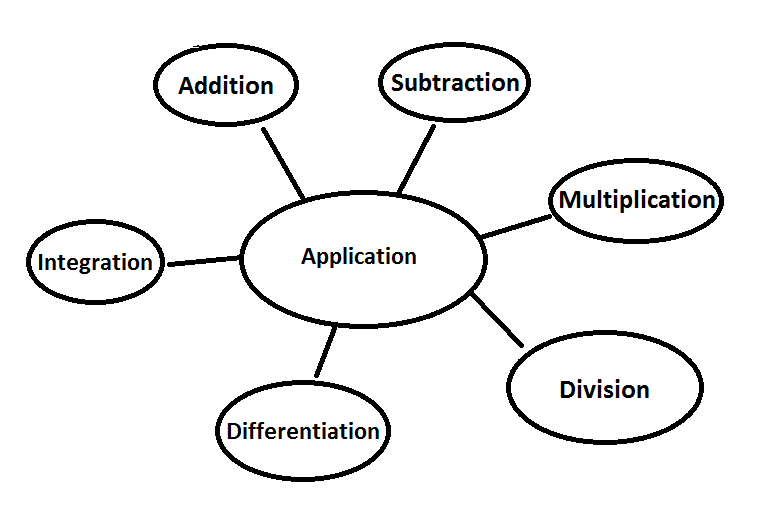
The corectness of the program does not impose any assumptions regarding the output of each operation. Although considering the freedom with which the user can enter the input, several assumptions can be made. Even though we assume that the user knows the correct format of the input, the expression is thoroughly examined before any operation is applied and so the user will be notified any time the input has an incorrect syntax and further information will be provided regarding the correct format. The same can be said for some of the operations. It is assumed that the user knows that the division by zero is not a valid operation but he will be notified in case such operation is tried. We also assume that the user does not need more than two decimals in the case of floating point numbers, therefore the result will be expressed with exact two decimals. Another assumption is that the degree of the monomials is positive.

b) Modeling

As mentioned above, an important aspect in finding a solution is to correctly represent the data. In the case of our problem, we chose to represent each monomial as an object of the class having the same name. Each monomial carries some characteristics such as the value of the coefficient, the value of the degree and some specific operations which can be applied on a monomial or between two monomials. The polynomial is expressed as a list of monomial object.

c) Use cases

The use-cases are represented by the functions the program offers: the capability of performing mathematical operations on polynomial expressions. All the available operations, i.e. use cases possible are shown in the following diagram:



For each of the above mentioned use-cases, a button on the graphical user interface will be present. In order for the user to perform an operation, he will first enter a polynomial expression having the established syntax and then will push the button corresponding to the desired use case and depending on the operation chosen, the result will be displayed (in the case of integration and differentation) or another polynomial expression is expected (in the case of addition, subtraction, multiplication and divisions, operations which involve two operands) followed by a click on the equal button to obtain the result. These steps are the same for all the use-cases the program should encounter.

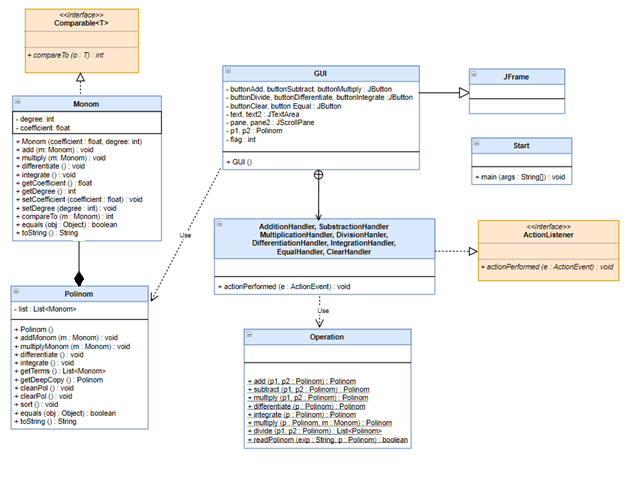
d) Errors

The corectness of the program guarantees no errors in terms of computing the result if the input is correctly entered. Even though the input entered may have a wrong format, the user will be immediately prompted and the program should not encounter any errors. At this point, it is safe to say that all the exceptions that might appear due to invalid input are addressed.

3. Design

The design of the application is made in an object oriented manner. As mentioned earlier, the polynomials itself are represented as lists of objects of type monomial. Monomial and polynomial are two different classes between which aggregation is established.

The following UML diagram shows the entire structure of the program along with all the classes, interfaces and relationships between them.



As we can clearly see from the UML diagram, the program has four main classes, namely Monom, Polinom, Operation and GUI. There are some relationships established between these classes. For example, we can see from the UML diagram that there is an aggregation between class Polinom and Monom, a relation of type “has-a” being established between them. We can also deduct from the diagram that there is a dependency on Polinom class in GUI and also the inner classes of GUI, which implements buttons handlings, depends on the class Operation. When it comes to inheritance and implementation of the interface, it’s noticeable that class GUI extends JFrame in order to create the application window. The classes responsible for event handlers of type ActionEvent also implements the interface ActionListener in order to define the functionality of the buttons when events occur. Class Monom implements the interface Comparable in order to sort the polynomial based on each monomial’s degree.

When it comes to data representation, an ArrayList of monomials was chosen to represent a polynomial expression. We chose this representation because the ArrayList offers some neat features extremely useful in this scenario. For example, the ArrayList does not have a fixed size and provides a set of useful methods for adding a monomial, deleting a specific element based on a certain property (in this case the monomials with coefficient 0), efficiently sorting the entries and even deleting the entire list of monomials.

All the main classes will be explained in detail in the following section along with their implementation, main methods and also the graphical user interface.

4. Implementation

We already mentioned that there are four main classes in this project. In what follows, we’ll explain each class in detail along with the attributes and methods contained.

Class Monom is the representation of a monomial. As already mentioned, a monomial is defined by its coefficiant (floating point number) and degree (integer). Along with these, class Monom has several other methods such as the one specialized on performing mathematical operations associated with monomials along with getters and setters. The classes specialized on performing operations simply apply the mathematical rules on the monomial’s coefficient and degree in order to obtain the desired result. There is also a compareTo method used to sort the polynomial based on each monomial’s degree in descending order.

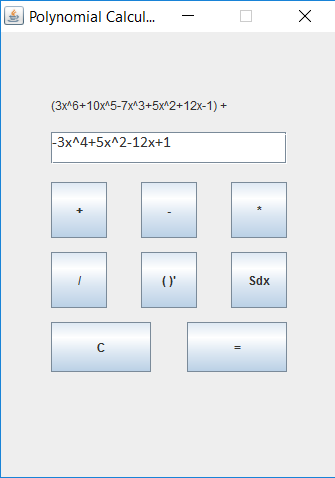
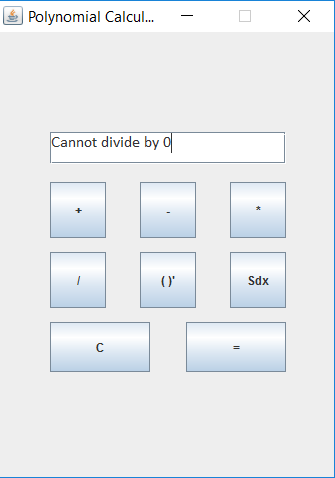
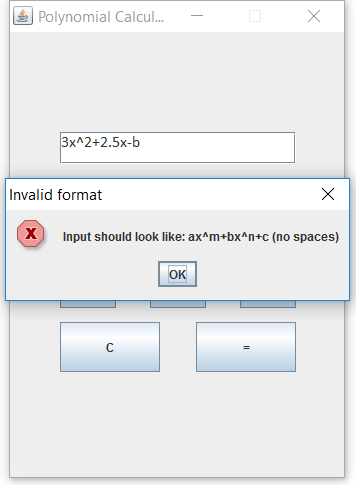
Class Polinom offers a representation of a polynomial expression along with all the methods applied on a polynomial or between two expressions. As a design choice, we chose to store the monomials of each polynomial in a ArrayList. The purpose of doing this was clearly explained in the design section. Besides the constructor which is used to initialize the list, this class contains some useful methods for processing polynomials. Method addMonom adds a new instance of Monom to the list if there is no other monomial having the same degree, otherwise adds their coefficiants. Method multiplyMonom multiplies a polynomial with a monomial by calling the multiply method from class Monomial for each monomial in the list. Similarly, methods differentiate and integrate calls the same methods from class Monomial for each element of the list. Method getDeepCopy creates a deep copy of a polynomial, by creating another object and duplicating the entries of the current one. Method cleanPol deletes from polynomial the monomials with coefficient 0 (used for display purpose) and clearPol clears the entire list of monomials associated with a polynomial.

Class Operation acts as a service class providing the functionalities of the program. Each method is declared static for obvious reasons and implements the operations specified in the beginning. The result is return as an object of type Polinom. Method add adds two polynomials by calling the method addMonom from class Polinom for each monomial in the second operand. Similarly, class subtract multiplies each monomial of the second operand with value -1 and adds the two polynomials, resulting in a subtraction. Method multiply multiplies two polynomials by calling the multiply method from class Polinom using the first operand for each monomial in the second polynomial and adding the result after each step into a new object of type Polinom. Differentiate and integrate are pretty straight forward, calling the functions with the same name from the parent class. Method divide can be seen as a bit more complex. The idea behind it is the folowing: as long as the degree of the first polynomial is equal or greater than the degree of the second polynomial and the absolute value of the coefficient of the element with the highest degree in the first polynomial is greater than the other polynomial’s in the same conditions, a new monomial can be added to the quotient. The new monomial’s coefficient will be equal to the coefficient of the first one divided by the coefficient of the second one and its degree will be equal to the result of subtracting the second polynomial degree from the first one. After that the newly obtained monomial will be multiplied with the divisor and the subtracted from the divident. When the condition will no longer be true the algorithm end and the quotient and remainder is obtained. Last but not least, the method readPolinom which extract the coefficient and the degree from a string representation of a polynomial for each monomial and adds them to an object of type Polinom. The function gets as parameters the string and an object of type Polinom where it stores the result. It uses a regex to split the given string into monomials but before doing that, it thoroughly check the string to make sure that it obeys the established format. If input string has an invalid format, the method will return false. It will return true if the string was correcly processed, i.e. has a valid format.

Class GUI is the one responsible with the drawing of the graphical user interface along with the button’s functionalities and error addressing. It consists of seven JButton instances specific for each type of operations performed by the program along with an equal and clear button. Besides these attributes, the class also contains two JTextArea instances, one for getting the text and printing the result and the other which is not editable (the user cannot enter input there) used to display intermidiate or additional information (the first operand or the remainder in case of division). There are also two JScrollPane instances which offer the scrolling capabilieties to the text area when the polynomial surpasses the length of the box. The flag variable is used to indicate the current operation being performed. In order to display the graphic components, class GUI extends JFrame. The entire class consists of a single method, the constructor, which initializes each component and places them in the application’s window. To offer the desired functionality to each button, a private class which implements ActionListener and defines the method actionPerformed has been created for each one.

All classes defined for operation buttons have similar functionalities and therefore not all of them will be described. The functionality is the following: when the users enters a polynomial in the text area and then presses the button corresponding to the desired operation, the method gets the text, first replaces all the capital X with x and then calls the method readPolinom from class Operation. If the method returns true, the input was valid, otherwise the user will be prompted by a warning message. If the text was valid, then that polynomial expression will be saved in object p1 of type Polinom and it will be displayed in the top JTextArea. The flag will be set corresponding to the chosen operation (1 for addition, 2 for subtraction, 3 for multiplication and 4 for division). After this, users must enter the second polynomial in the case the operation requires two operands. After the second polynomial was entered, the user must press the equal button. The class which implements the functionality for the equal button checks the flag to see what operation must be performed, stores the representation in the object p2 of type Polinom and then calls the desired method from the Operation class to obtain the result. If the second polynomial has an invalid syntax, the user will be again prompted by a warning message. When integration and differentiation is performed, the user does not need the equal button anymore because the result will be displayed immediately after the button corresponding to those operations will be pressed.

All the errors that might appear due to invalid input are addresed inside this class. For example, each implementation of actionPerformed checks if the readPolinom method gets a valid input, otherwise the user will be notified. When the user tries to divide a polynomial by 0 another message will appear and notify that the disired operation cannot be performed. The following images shows the graphica user interface along with the error messages.

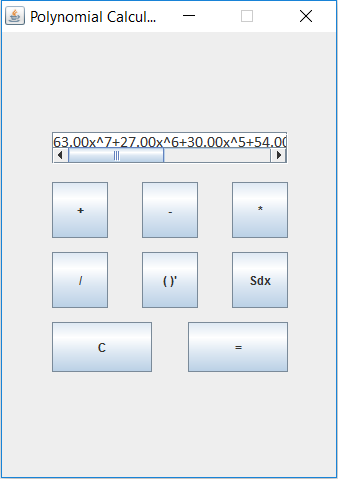
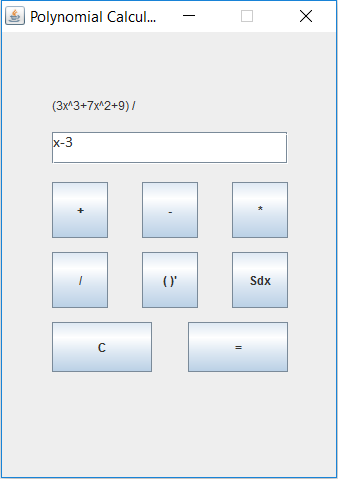
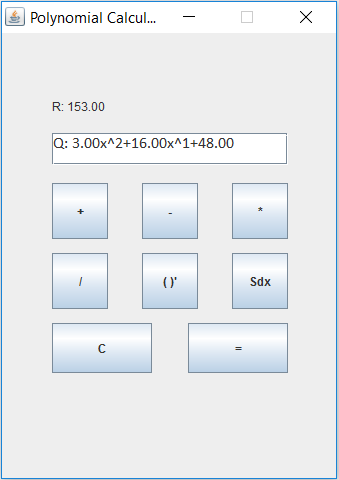
5. Testing

In order to make sure that the program behaves correctly and outputs the desired result, testing must be conducted. In this case, JUnit test framework was used to show the corectness of the program. The components tested were the operation implementation on polynomial expression. Therefore, for each operation two polynomials were entered (one in case of differentiation and integration) and then checked for equality with another polynomial which stores the correct result. The input for each operation tested as well as the expected and the obtained result are shown in the following table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operation | Input | Expected output | Obtained output | Result |
| Addition | + | () |  | PASSED |
| Subtraction |  |  |  | PASSED |
| Multiplication |  |  |  | PASSED |
| Differentiation |  | () | () | PASSED |
| Integration |  |  |  | PASSED |

6. Results

After all the necessary steps (design, implementation, testing) we managed to obtain a fully functional polynomial calculator which is able to perform various operations on polynomial expressions. Also, besides the main functionality, a simple and intuitive graphical user interface was created to ease the access to the program’s functions. The following screenshots present the application in various use cases.

7. Conclusions

During the design and implementation of this project I was able to recap the knwoledge regarding mathematical operations applied on polynomial expressions. Moreover and more importantly, my Java programming skills have been improved. Although the problem was not extremely hard, some design choices needed to be made and overall I managed to learn some new things such as using regular expressions to split strings of characters.

When it comes to future improvements of the project there are some nice features I would suggest. For example, a graphical representation of the polynomial function can be implemented. Another addition would be the support for polynomial expressions with multiple variables and the ability to compute the value of a polynomial in a given point.

8. Biobliography

<https://stackoverflow.com> (for various implementation questions)

<https://stackoverflow.com/questions/36490757/regex-for-polynomial-expression> (regex for polynomials)

<https://www.draw.io/> (for drawing the UML diagram)

Java: How to Program, 9th Edition (Deitel) (for Java, GUI and UML recap)